3

Working with MySQL

After installing and initializing MySQL, you can begin working with the server and client tools that are included in that installation. Before you begin creating databases and tables, inserting and manipulating data, or modifying server and client configurations, you should have a basic understanding of how to use MySQL. This includes not only finding your way through the MySQL directory structure, but also knowing what server-related and client programs are included in the MySQL installation and what steps you can take to control how those programs run. In addition, you must know how to protect your installation so that only those users that you want to have access to MySQL are permitted access.

To prepare you to start using MySQL, this chapter covers many of the topics that can give you the foundation necessary to work with MySQL. The chapter contains details about the directory structure set up when you install MySQL, the programs available as part of that installation, and the methods used to configure the options available to those programs. In addition, the chapter includes information about how to secure MySQL so that you can prevent access by unauthorized users. Specifically, the chapter covers the following topics:

- The MySQL directory structure that is used in Linux RPM and tar installations and in Windows installations. In addition, you'll learn about the data directory and the grant tables, as they're implemented in Linux and in Windows.
- Information about MySQL server-related and client programs. You'll learn how to specify program options at the command prompt and in a configuration file, and you will be instructed on how to use the mysql client utility.
- How to assign passwords to MySQL user accounts.

Understanding the MySQL Directory Structure

When you install MySQL, a directory structure is set up to support the various database-related functions. The directories contain the files necessary to initialize the database, start the MySQL server, and set up the server to run automatically. In addition, the directories include the

server-related and client programs included with MySQL, as well as script, log, and document files related to the MySQL operation. This section discusses the details of the directory structure and then focuses specifically on the data directory, which houses the actual database files.

MySQL File Storage

The way in which the MySQL directories are structured can vary depending on the operating system on which MySQL is installed and the distribution type used to perform that installation. Because Chapter 2 explained how to install MySQL on Linux (using RPM and tar files) and on Windows, these are the three directory structures that are reviewed here. For each installation type, you are given a table that outlines how the MySQL files are stored. In addition to reviewing the appropriate table, be sure to view the contents in each directory so that you can become familiar with where the files are located. As you work your way through the book, you can refer back to the table to determine where a file is located.

File Storage for a Linux RPM Installation

As you'll recall from Chapter 2, when you execute the RPM installation files on your Linux computer, the MySQL files are extracted to your hard disk, the databases are initialized, and the server is set up to start automatically. When many of the files are extracted to your hard disk, they are distributed among directories that already exist in the Linux structure. In addition, several directories specific to MySQL are created, although most of the distributed files, as shown in the following table, are extracted to existing directories.

Directory	Contents
/usr/bin	Contains many of the binary program files and script files that you need to set up and interact with MySQL, including mysql, mysqladmin, mysqld_safe, and mysql_install_db. The /usr/bin directory is usually the first place you should look if you're trying to locate a MySQL program. Note that this directory also contains numerous programs and scripts that are unrelated to MySQL.
/usr/lib/mysql	Contains a symbol file named mysqld.sym that is used to troubleshoot MySQL in case it fails.
/usr/sbin	Contains the mysqld program file for the MySQL server.
/usr/share/doc/packages/MySQL-server	Contains the MySQL users' manual in .html and .txt file formats. The directory also contains the five sample configuration files: my-huge.cnf, my-innodb-heavy-4G.cnf, my-large.cnf, my-medium.cnf, and my-small.cnf.
/usr/share/info	Contains the MySQL manual in a compressed format (mysql.info.gz). A .gz file is a file that has been compressed with the Gzip utility.
/usr/share/man/man1	Contains manual pages that provide information about individual MySQL programs, such as mysql, mysqladmin, and mysqld.
/usr/share/mysql	Contains error messages for the different languages supported by MySQL. For each of these languages, an errmsg.sys and errmsg.txt file are provided. The directory also contains miscellaneous script and support files.
/var/lib/mysql	Contains log files as well as directories and files related to MySQL databases.

As you can see, an RPM installation spreads the files out into a number of directories. Unlike other distribution types, which tend to consolidate the MySQL files into a central location, the RPM distribution uses a less intuitive approach to file organization, so it can sometimes take a while to locate a specific file. The location that you're likely to be the most concerned with as you're learning about MySQL is the `/usr/bin` directory, which contains the majority of the program files used to run, access, and configure MySQL. Although you need to access other directories when using MySQL, you launch most of your MySQL programs from the `/usr/bin` directory.

Keep in mind that an RPM installation is based on individual RPM files. The preceding table reflects only the server and client installations. If you install additional RPM packages, your directory structure will reflect those installations.

File Storage for a Linux Tar Installation

A tar installation, although more complicated to implement than an RPM installation, organizes the MySQL files in a far simpler structure, which often makes locating files easier and quicker. As the following table demonstrates, you can access all the MySQL files through the `/usr/local/mysql` directory.

Directory	Contents
/usr/local/mysql/bin	Contains binary program files such as mysql, mysqld, and mysqld_safe. The /usr/local/mysql/bin directory is usually the first place you should look if you're trying to locate a MySQL program.
/usr/local/mysql/data	Contains log files as well as directories and files related to MySQL databases.
/usr/local/mysql/docs	Contains the MySQL users' manual in .html and .txt file formats. The directory also contains the manual saved as a mysql.info file.
/usr/local/mysql/include	Contains MySQL include files, such as config.h, hash.h, and my_dir.h.
/usr/local/mysql/lib	Contains compiled library files, such as libdbug.a and libmysqld.a.
/usr/local/mysql/man	Contains manual pages that provide information about individual MySQL programs, such as mysql, mysqladmin, and mysqld.
/usr/local/mysql/mysql-test	Contains the MySQL Test Suite, which allows you to perform regression tests on the MySQL code.
/usr/local/mysql/scripts	Contains the mysql_install_db initialization script.
/usr/local/mysql/share	Contains error messages for the different languages supported by MySQL. For each of these languages, an errmsg.sys file and an errmsg.txt file are provided.
/usr/local/mysql/sql-bench	Contains the files necessary to perform benchmark tests.

Table continued on following page

Directory	Contents
/usr/local/mysql/support-files	Contains the five sample configuration files: my-huge.cnf, my-innodb-heavy-4G.cnf, my-large.cnf, my-medium.cnf, and my-small.cnf. The directory also contains scripts, including mysql.server.

As you can see from the table, the directory structure is far simpler than that found in an RPM installation. In fact, for this reason alone you might consider a tar installation rather than an RPM installation. As you found with the RPM installation, the tar directory structure also includes a single directory that contains most of the MySQL programs. This directory — `/usr/local/mysql/bin` — is the one that you access the most as you're learning about MySQL, although you'll have to access other directories for files such as script files and sample configuration files.

File Storage for a Windows Installation

The directory structure in a Windows installation is similar to what you find in a tar installation on Linux. The MySQL files are all organized in a common directory structure under the `C:\Program Files\MySQL\MySQL Server <version>` directory, as shown in the following table.

Directory	Contents
C:\Program Files\MySQL\ MySQL Server <version>\	Contains the six sample configuration files: my-huge.ini, my-innodb-heavy-4G.ini, my-large.ini, my-medium.ini, my-small.ini, and my-template.ini. In addition, the initial configuration file (my.ini) created by the MySQL Server Instance Configuration Wizard is placed in this directory. The directory also contains files that provide licensing information.
C:\Program Files\MySQL\ MySQL Server <version>\bin	Contains binary program files such as mysql.exe, mysqld-nt.exe, and mysqladmin.exe. This is usually the first place you should look if you're trying to locate a MySQL program.
C:\Program Files\MySQL\ MySQL Server <version>\data	Contains log files as well as directories and files related to MySQL databases.
C:\Program Files\MySQL\ MySQL Server <version>	Contains the MySQL users' manual in .html and .txt file formats. This directory is included only for a .zip file installation, not a .msi file installation.
C:\Program Files\MySQL\ MySQL Server <version>\share	Contains error messages for the different languages supported by MySQL. For each of these languages, an errmsg.sys file and an errmsg.txt file are provided.

In Windows, the MySQL program files are located in the `C:\Program Files\MySQL\MySQL Server <version>\bin` directory. Although you have to access files from other directories, the `C:\Program Files\MySQL\MySQL Server <version> \bin` directory is the one that you access the most often as you're learning to use MySQL.

The Data Directory

Now that you have an overview of the directory structures for the three installation types discussed so far, you can take a closer look at the data directory, which contains the database files used to support the MySQL databases. The data directory also contains log files related to those databases.

As you saw earlier in the chapter, the location of your data folder depends on your computer's operating system and on the distribution method you used to install MySQL. For the three platforms and installation types discussed, the data directories are located as follows:

- For an RPM installation on Linux: `/var/lib/mysql`
- For a tar file installation on Linux: `/usr/local/mysql/data`
- For a Windows installation: `C:\Program Files\MySQL\MySQL Server <version>\data` `C:\mysql\data`

In addition to log files, the data directory contains a subdirectory for each database that exists in MySQL. The database subdirectories share the same names as the databases themselves. For example, suppose you add a database named TestDB to your MySQL installation. In Windows, the directory associated with the TestDB database is `C:\Program Files\MySQL\MySQL Server <version> \data\TestDB`. In an RPM installation of Linux, the directory associated with the TestDB database is `/var/lib/mysql/TestDB`. In a tar installation, the directory is `/usr/local/mysql/data/TestDB`.

When you first install MySQL and initialize that installation, the following two databases are created by default:

- **mysql:** An administrative database that contains the system tables necessary to control user access, provide help-related information, and support time-zone-related functionality.
- **test:** A sample database that you can use to test MySQL functionality. The database contains no tables, although you can add tables as necessary.

As a result of there being two default databases included in your MySQL installation, there are two default subdirectories: `mysql` and `test`. Each database subdirectory contains files that map to any tables that exist in that database. Because the test database contains no tables, there are no files associated with that database. If you were to add tables, the necessary files would be added to the test subdirectory. Because the mysql database contains tables, the `mysql` subdirectory contains numerous files.

Whenever you add a table to a database, one or more of the following types of files are created in the database subdirectory:

- **.frm:** The primary table-related file that is used to define the table's format. All table types have an associated .frm file.
- **.MYD:** The file that stores the data contained in several table types.
- **.MYI:** An index file used by several table types.
- **.MRG:** A special type of file that is used to list the names of merged tables.

Which files are created for a table depends on the table's type. The following table provides a brief description of each table type supported by MySQL and lists the files that are created when you add a table to a database.

Table type	Description	Files used
BDB	A transaction-safe table that the Berkeley DB handler manages. For the most part, InnoDB tables have replaced BDB tables.	.frm, .MYD, .MYI
MEMORY	A table whose contents are stored in memory. The data stored in the tables is available only as long as the MySQL server is available. If the server crashes or is shut down, the data disappears.	.frm
InnoDB	A transaction-safe table that the InnoDB handler manages. As a result, data is not stored in a .MYD file, but instead is managed in the InnoDB tablespace.	.frm
ISAM	A deprecated table type that was once the default table type in MySQL. The MyISAM table type has replaced it, although it is still supported for backward compatibility.	.frm, .MYD, .MYI
MERGE	A virtual table that is made up of multiple MyISAM tables. Data is not stored in the MERGE table, but rather in the underlying MyISAM tables.	.frm, .MRG
MyISAM	The default table type in MySQL. MyISAM tables, which have replaced ISAM tables, support extensive indexing and are optimized for compressions and speed.	.frm, .MYD, .MYI

You define the table type when you create a table. If you specify no table type , a MyISAM table is created by default. For more information about table types and how they apply to table definitions, see Chapter 5.

The files created for a table that is added to a MySQL database share the same name as that table. For example, the mysql database includes a table named user. Because the user table is a MyISAM table type, three files exist for that table — user.frm, user.MYD, and user.MYI — and the files are stored in the mysql subdirectory. For instance, in Windows, the user.frm file is located at `C:\Program Files\MySQL\MySQL Server <version>\data\mysql\user.frm`; in an RPM installation on Linux, the file is located at `/var/lib/mysql/mysql/user.frm`; and in a tar installation on Linux, the file is located at `/usr/local/mysql/data/mysql/user.frm`.

The mysql Database

The mysql database is an administrative database that contains tables related to securing the MySQL installation, storing user-defined functions, and providing data related to the MySQL help system and to time-zone functionality. The mysql database must be initialized before you can start using MySQL. When you use RPM files to install MySQL on Linux or you install MySQL on Windows, the mysql database is initialized by default. If you use a tar file to initialize MySQL, you must run the `mysql_install_db` script. (See Chapter 2 for details about running the `mysql_install_db` script.)

By default, the mysql database includes 15 tables. The following table provides a brief description of the data included in each table.

Table	Contents
columns_priv	Contains access-control data for individual columns within a specified table.
db	Contains access-control data that defines the type of privileges a user is granted on a specified database.
func	Contains data about user-defined functions that have been added to MySQL.
help_category, help_keyword, help_relation, help_topic	Contains data related to the MySQL help system. There are four help-related tables in all.
host	Contains access-control data that defines the type of privileges a host is granted on a specified database.
tables_priv	Contains access-control data for individual tables in a specified database.
time_zone, time_zone_leap_second, time_zone_name, time_zone_transition, time_zone_transition_type	Contains data related to time-zone functionality in MySQL. There are five tables related to time-zone functionality.
user	Contains access-control data that defines which users can connect to the MySQL server, from which computers those users can access MySQL, and the type of global privileges that the users have in order to access MySQL and its databases.

When you first begin using MySQL, you're most concerned with the tables that determine the privileges available to anyone who connects to MySQL. These tables, referred to as grant tables, are discussed in the following section.

The Grant Tables

A *grant table* is any one of the tables in the mysql database that are used to control access to MySQL and the MySQL databases. By default, MySQL creates the following five grant tables:

- columns_priv
- db
- host
- tables_priv
- user

The grant tables define which users can access MySQL, from which computers that access is supported, what actions those users can perform, and on which objects those actions can be performed. For example, the grant tables allow you to specify which users can view data in a particular database and which users can actually update that data.

The actions that users are permitted to take and the data that they are permitted to access are controlled through a set of privileges. The following table lists each of the privileges available in MySQL and the actions that they permit a user to take.

Privilege	Allows user to:
Select_priv	Query data in a database.
Insert_priv	Insert data into a database.
Update_priv	Update data in a database.
Delete_priv	Delete data from a database.
Create_priv	Create a table in a database.
Drop_priv	Remove a table from a database.
Reload_priv	Reload the data in the grant tables into MySQL.
Shutdown_priv	Shut down the MySQL server.
Process_priv	View a list of MySQL processes.
File_priv	Export data from a database into a file.
Grant_priv	Grant privileges on database objects.
References_priv	Functionality not yet supported, but the intention of this privilege appears to be to allow a user to configure foreign key constraints.
Index_priv	Create and delete indexes in a database.
Alter_priv	Alter database objects.
Show_db_priv	View all databases.
Super_priv	Perform advanced administrative tasks.
Create_tmp_table_priv	Create temporary tables.
Lock_tables_priv	Place locks on tables.
Execute_priv	Run a stored procedure. Functionality will not be supported until MySQL version 5.0.
Repl_slave_priv	Read binary logs for a replication master.
Repl_client_priv	Request information about slave and master servers used for replication.
Table_priv	Access a specified table in a database.
Column_priv	Access a specified column in a table in a database.

When you first install MySQL, a number of privileges are configured by default. The privileges are configured in the grant tables, as they exist in the mysql database. By default, no privileges are assigned in the columns_priv, tables_priv, and host tables. In fact, these three tables are empty; however, the user table and db table contain users that have been granted privileges. For that reason, the following sections describe the user and db tables in more detail.

Chapter 14 discusses the grant tables and user privileges in much more detail. The information is presented here only as a way to give you an understanding of the mysql database, how users access MySQL, and what you can do to prevent unauthorized access.

The user Table

The user table is the primary grant table in the mysql database. The table controls who can connect to MySQL, from which hosts they can connect, and what superuser privileges they have. A *superuser* privilege applies globally to MySQL. A user assigned a superuser privilege can perform the task defined by that privilege on any database in the system. Every MySQL user is listed in the user table, whether or not they are assigned privileges in that table. The user table provides the widest scope in a MySQL implementation, followed by the db and host tables. If a user is not listed in the user table, that user cannot connect to MySQL.

The user table is configured differently for a Linux installation than it is for a Windows installation. Consequently, this section looks at the table as it appears in each implementation.

Linux Implementation

When you install and initialize MySQL on Linux, entries are added to the user table for the root user and for anonymous users. The root user in MySQL is not the same as the root user in Linux. The root user in MySQL is specific to MySQL. By default, the user table contains two records for the MySQL root user and two records for anonymous users, as shown in Figure 3-1.

host	user	password	privileges
localhost	root		all privileges
<local computer name>	root		all privileges
<local computer name>			no privileges
localhost			no privileges

Figure 3-1

The figure represents how the user table is configured in MySQL. The privileges column is actually a consolidation of all the privilege-related columns in that table. For example, in the first record, the root user has been granted all privileges from the localhost, and in the second record, the same user has been granted all privileges from the local computer (represented by the `<local computer name>` placeholder) where MySQL is installed. Essentially, the first two records are the same. However, by creating one record for localhost and one record for the actual computer name, you can use either name as a program option.

The table also includes two records for anonymous users. Anonymous users are shown in the user table by a blank value in the user column. Unlike the root user, the anonymous users have not been granted any privileges. Like the root user, however, anonymous users do not have to provide a password. As a result, the default configuration of the user table provides the following access to the users specified in the table:

- The root user has superuser access that allows that account to perform all administrative tasks. This access, however, is from the local computer only.
- No password is assigned to the root user, so anyone can sign in as the root user from the local computer and have full administrative access to MySQL.
- Anonymous users can connect to MySQL from the local computer, although they are denied superuser access.

As you can see, this is not a secure installation. It is up to you to determine whether this default configuration is adequate or whether you need to restrict access. It's generally not a good idea, though, to leave access as open-ended as it is here.

Windows Implementation

Like the Linux installation, entries have been added to the user table for the root user, as shown in Figure 3-2. The user table, though, is configured slightly differently in a Windows environment. (Again, the privileges column is merely a consolidation of all the privilege-related columns in the user table.) Notice that, in Windows, the root user can connect to MySQL from any host, as the percentage (%) placeholder represents. In Linux, the root user can connect only from the local computer.

host	user	password	privileges
localhost	root	*AA25B3745CB38F87F8BB4C12F28200463FC2D2E3	all privileges
%	root	*AA25B3745CB38F87F8BB4C12F28200463FC2D2E3	all privileges

Figure 3-2

Another difference between the Windows installation and the Linux installation is that, in Windows, each instance of the root user is assigned a password, shown as an encrypted value in the table. This is the password that you assigned to the root user when you ran the MySQL Server Instance Configuration Wizard. As with the Linux root user, the Windows root user is assigned all privileges. As a result, the root user has superuser access that allows root to perform all administrative tasks — from any computer.

As the user table shows, a Windows installation is, initially, more secure than a Linux installation. Later in the chapter, you will learn how to secure your Linux installation so that access is more restrictive than it is by default.

The db table

The db table is configured initially to allow access to anonymous users. The db table defines which users can connect to which databases from which hosts. The db table is configured the same in Linux and Windows installations, as shown in Figure 3-3.

host	db	user	privileges
%	test		all privileges except those that allow the user to grant privileges
%	test_%		all privileges except those that allow the user to grant privileges

Figure 3-3

The db table includes two rows, both for anonymous users, and both rows allow the users to connect from any host. As a result, any user can access the test database from any host. In addition, any user can access any database that begins with `test_` from any host. (The backslash [\] that precedes the underscore indicates that the underscore should be used as a literal character. The percentage sign [%] indicates that any value can follow the underscore.) For users to be granted the access that is configured in the db table, they must exist as users in the user table.

Using the MySQL Programs

Now that you have an overview of how directories are structured in MySQL, take a look at the programs included in a MySQL installation. MySQL includes programs that are related to running the MySQL server or that are client tools used to interact with the server. For all the MySQL programs, you can specify a variety of options that affect how that program runs and the context in which it operates.

Specifying Program Options

Most MySQL programs support numerous options that you can specify when you invoke the program. The number of options that you might choose to include with one program might become a bit unwieldy, especially if you have to retype those options over and over again. As a result, MySQL supports a variety of ways to specify options available to your programs:

- **Command prompt entries:** You can specify options and, when applicable, their values when you type the command at a command prompt.
- **Configuration files:** You can add options to a configuration file that are specific to a program or available to all client programs. For example, you can specify the user and password to use for a specific program.
- **Aliases:** If your operating system supports the creation of aliases, you can create an alias whose definition includes specific options that you want to set.
- **Scripts:** You can create a shell script that defines the program and its options, and then you can call that script from within your shell.
- **Environment variables:** You can set environment variables that affect MySQL program operations. For example, you can use the `MYSQL_HOST` environment variable to define the name of a host to connect to when connecting to a MySQL server.

Of the methods available for specifying program options, using command prompt entries and using configuration files are the most common. For that reason, these are the two methods focused on in this chapter. For more information about creating scripts and aliases, see your operating system's documentation. For more information about environment variables, see the MySQL product documentation.

Specifying Options at a Command Prompt

The first step in specifying options for any program is to know what options are available for that program. The easiest way to determine what options are available is to use the help option. To use the help option, simply type the name of the program at your shell's command prompt and add `--help`, as shown in the following command:

```
mysqlaccess --help
```

When you execute this statement, information about the mysqlaccess program displays. This information, which includes details about which options are available and what those options are used for, displays in your shell. You can also access help by using a shorter version of the command:

```
mysqlaccess -?
```

This command returns the same results as `--help`.

Most programs include a long and a short name for each option. In addition, for many of the options, you must specify values. For example, suppose you want to operate the mysql client utility in the context of the root user. You would type the following command:

```
mysql --user=root
```

When mysql launches, it runs under the context of the specified user. You could have also specified the option by using the short name for that option, as in the following command:

```
mysql -u root
```

Notice that, when specifying the short name, you need to use only a single dash, rather than two, and you do not use the equal sign (=). This is typical for most MySQL programs. In addition, if you use the short name for the option, you do not have to separate the option name from the values. For example, you can use the `-u` option in the following manner:

```
mysql -uroot
```

As you can see, there is no space between the `-u` and the word root. For most options, you can specify the argument with or without a space. For the short name of the password option (`-p`), though, you can never add a space. The password, when specified, must always follow immediately after the `-p`. But this is generally not a concern because you normally use the password option without specifying the actual password until after you press Enter. As a result, your command would look like the following:

```
mysql -u root -p
```

When you press Enter, you're prompted for the password. At the prompt, you enter the password and then press Enter again.

Not all options require the use of an option name to precede its value. For example, if you want to specify a database when you start the mysql utility, you simply type the database name, after the name of the program, as in the following command:

```
mysql test -u root -p
```

When mysql launches, it prompts you for a password and then opens in the context of the root user account and the test database.

You will find, as you're working with various MySQL programs, that a number of options are common to those programs. You already saw examples of some of them, such as `--help` (`-?`), `--host` (`-h`), `--user` (`-u`), and `--password` (`-p`). In addition to these common options, each program includes options specific to the program. In some cases, the options that are available to a specific program can depend on the operating system being used and the edition of MySQL.

You might discover when you're entering commands at a command prompt that you're repeating the same commands over and over, which can become tedious very quickly. One way around this repetition is to use the functionality available at your particular command prompt that allows you to repeat commands. For example, you might be able to use the Up arrow to repeat a previous command. Be sure to check the documentation for your shell window to determine how to repeat commands.

You also might want to consider modifying the `PATH` *environment variable for your operating system so you don't have to type in a complete path every time you want to start a MySQL program. For example, you could modify the* `root/.bash_profile` *file on a Linux tar installation by adding* `:/usr/local/mysql/bin` *to the* `PATH` *environment variable. (The colon separates the path from the previous entry.) Because the RPM MySQL installation on Linux installs many of the MySQL programs in* `/usr/bin`*, you might not need to modify the* `PATH` *environment variable because* `/usr/bin` *is already a supported path for some versions of Linux.*

Modifying environment variables in Windows is a little different from Linux. For Windows, you can use the System utility in Control Panel to modify the environment variable. To change the variable, open the System utility, click the Environment Variables button on the Advanced tab, and then edit the PATH variable by adding `;C:\Program Files\MySQL\MySQL Server <version> \bin` *to the path list. (The semi-colon separates the path from the previous entry.) You can also use the set command from a Command Prompt window to set the PATH environment variable.*

If you modify an environment variable, be sure to close the command shell and then reopen it to make sure that the new path specification takes effect.

Specifying Options in a Configuration File

A *configuration file* (or *option file*) is referenced by a number of MySQL programs when those programs are launched. The configuration files contain settings that are comparable to the options that you can specify at a command line when you run the program. Any options available at a command line are available to use in a configuration file. You can use only the long-name version of the option, and you don't use the dashes. For example, suppose you want to include the `user` option in a configuration file. If you were going to specify that option at a command prompt, you would enter it as follows:

```
mysqladmin --user=root
```

If you were going to include this option in a configuration file, you would type the following:

```
user=root
```

The option is added beneath the program heading, which in this case is `[mysqladmin]`. You could then add more options on the lines beneath the one you just entered. Unlike using options on a command line, place the options in a configuration file on separate lines, as in the following:

```
[mysqladmin]
host=server12
user=root
user=pw1
```

Any option placed under the `[mysqladmin]` heading is used when you run the mysqladmin program. You can override these options by specifying the option at a command prompt when you run the program.

Both Linux and Windows (as well as other operating systems) support configuration files. On Linux, the name of the configuration file is `my.cnf`, and you can place it in any of the following locations, depending on the intended scope of the file:

- **Global options:** `/etc/my.cnf`
- **Server-specific options:** `<data directory>/my.cnf`
- **User-specific options:** `~/.my.cnf`

On Windows, the name of the configuration file is `my.ini`. When you run the MySQL Server Instance Configuration Wizard (right after you install MySQL), the wizard creates a `my.ini` file and places it in the `C:\Program Files\MySQL\MySQL Server <version>` directory. By default, this is the location that the MySQL service looks for the `my.ini` file when it starts the MySQL server. The MySQL client tools, however, look for the `my.ini` file in the `C:\WINDOWS` directory. As a result, you must maintain two separate configuration files or re-create the MySQL service so that it points to the `C:\WINDOWS` directory. Regardless of the location of the option file, the options specified are global.

> *To re-create the MySQL service, you must stop the service, remove the service, create a new service, and then start that service. The new service should point to the mysqld-nt.exe program file in the* `C:\Program Files\MySQL\MySQL Server <version>\bin` *directory and the my.ini configuration file in the* `C:\WINDOWS` *directory. In Chapter 13 you will learn how to re-create the service when you implement logging. For details about Windows services, see the Windows product documentation.*

In any configuration file, the options are grouped according to a program or type of program. Each grouping is preceded by a program heading, which is the name of the program in brackets, as you saw previously with `[mysqladmin]`. You can create a configuration file from scratch or copy one of the sample configuration files. If you're working in Windows, you can also copy the `my.ini` file from the `C:\Program Files\MySQL\MySQL Server <version>` directory to the `C:\WINDOWS` directory.

In Linux, MySQL includes five sample configuration files: `my-huge.cnf`, `my-innodb-heavy-4G.cnf`, `my-large.cnf`, `my-medium.cnf`, and `my-small.cnf`. Windows includes the same five (but with an .ini extension rather than .cnf), plus an additional template file named `my-template.ini`. The differences among these files have mostly to do with the sizes of the cache and buffer settings. To determine which configuration file might be best for your use, take a look at the contents of each file and decide, based on your system's configuration, which one is best for you.

Each sample configuration file provides program headers for a number of the programs available in MySQL. Much of the content in these files are comments, which are indicated by the number sign (#) at the beginning of a line. The program ignores comments. For example, the following code is from the `my-small.ini` configuration file available in a Windows installation:

```
# Example MySQL config file for small systems.
#
# This is for a system with little memory (<= 64M) where MySQL is only used
# from time to time and it's important that the mysqld daemon
# doesn't use much resources.
#
# You can copy this file to
# /etc/my.cnf to set global options,
# mysql-data-dir/my.cnf to set server-specific options (in this
# installation this directory is /usr/local/mysql/var) or
# ~/.my.cnf to set user-specific options.
#
# In this file, you can use all long options that a program supports.
# If you want to know which options a program supports, run the program
# with the "--help" option.

# The following options will be passed to all MySQL clients
[client]
#password       = your_password
port            = 3306
socket          = /tmp/mysql.sock
```

```
# Here follows entries for some specific programs

# The MySQL server
[mysqld]
port            = 3306
socket          = /tmp/mysql.sock
skip-locking
key_buffer = 16K
max_allowed_packet = 1M
table_cache = 4
sort_buffer_size = 64K
read_buffer_size = 256K
read_rnd_buffer_size = 256K
net_buffer_length = 2K
thread_stack = 64K

# Don't listen on a TCP/IP port at all. This can be a security enhancement,
# if all processes that need to connect to mysqld run on the same host.
# All interaction with mysqld must be made via Unix sockets or named pipes.
# Note that using this option without enabling named pipes on Windows
# (using the "enable-named-pipe" option) will render mysqld useless!
#
#skip-networking
server-id       = 1

# Uncomment the following if you want to log updates
#log-bin

# Uncomment the following if you are NOT using BDB tables
#skip-bdb

# Uncomment the following if you are using InnoDB tables
#innodb_data_home_dir = /usr/local/mysql/var/
#innodb_data_file_path = ibdata1:10M:autoextend
#innodb_log_group_home_dir = /usr/local/mysql/var/
#innodb_log_arch_dir = /usr/local/mysql/var/
# You can set .._buffer_pool_size up to 50 - 80 %
# of RAM but beware of setting memory usage too high
#innodb_buffer_pool_size = 16M
#innodb_additional_mem_pool_size = 2M
# Set .._log_file_size to 25 % of buffer pool size
#innodb_log_file_size = 5M
#innodb_log_buffer_size = 8M
#innodb_flush_log_at_trx_commit = 1
#innodb_lock_wait_timeout = 50

[mysqldump]
quick
max_allowed_packet = 16M

[mysql]
no-auto-rehash
# Remove the next comment character if you are not familiar with SQL
#safe-updates
```

```
[isamchk]
key_buffer = 8M
sort_buffer_size = 8M

[myisamchk]
key_buffer = 8M
sort_buffer_size = 8M

[mysqlhotcopy]
interactive-timeout
```

Notice that most of the lines in this file are comments, which means that the program disregards these lines when it is launched. The comments serve as guidelines. You can use the settings suggested in the comments or define the options as best fit your needs.

One section of the configuration file that you should be aware of is the one that uses the `[client]` heading. Any options specified in this section apply to all MySQL client programs. Whenever you start one of the client programs, it checks the section specific to the program and also checks the `[client]` section.

As you can see, the configuration files are a handy way to specify your program's options. This is especially useful for those programs that require multiple options or that you invoke over and over again. With a configuration file, all you need to specify is the name of the program at a command prompt; the options are applied automatically.

Server-Related Programs, Scripts, and Library Files

MySQL includes a number of programs, scripts, and library files related to the operation of the server. The following table provides a description of each of these. To learn which options are available for each one, type the filename, along with the `--help` option, at a command prompt, and then press Enter.

Server-related file	Description
libmysqld	Library file that is used to embed the MySQL server into other applications. The libmysqld file is not actually a program, but it can be used with other stand-alone programs so that they can include a MySQL server.
mysql.server	Script file that you can use on Unix-like systems to start and stop the MySQL server automatically. The script invokes the mysqld_safe startup program file. If you use an RPM file to install the MySQL server on a Linux computer, the mysql.server script is implemented automatically.
mysql_install_db	Script file that creates and populates the initial databases (mysql and test) after MySQL has been installed. If you perform a tar-based installation of MySQL, you must run the mysql_install_db script manually to initialize the databases.
mysqld	MySQL server program file. The mysqld program must be running to support client connections because access to the data is through the server. Windows installations include versions of this program that are optimized for Windows.

Server-related file	Description
mysqld-max	MySQL server program file that includes features in addition to what's found in the standard mysqld program file. The mysqld-max program must be running to support client connections because access to the data is through the server. Windows installations include versions of this program that are optimized for Windows.
mysqld_multi	Script file that you can use to manage multiple mysqld processes. The script can start and stop the servers as well as report their current status.
mysqld_safe	Script file that starts the MySQL server automatically, restarts it if necessary, and monitors it. Using the mysqld_safe script is the recommended way to start MySQL.

Client Programs

The MySQL client programs allow you to interact with the MySQL server and the data stored in MySQL. The following table describes many of the client programs available in MySQL.

Client program	Description
myisamchk	Checks and repairs MyISAM tables. The MyISAM format is the default table type in MySQL and provides the basic functionality you would expect from a table in a database. The myisamchk utility is an updated version of the isamchk utility, which is used for ISAM tables. You should not use the myisamchk utility when the server is running.
myisampack	Compresses MyISAM tables into read-only tables in order to reduce storage requirements. The myisampack utility is an updated version of the pack_isam utility.
mysql	Supports access to the data in a MySQL database. You can use the utility in interactive mode or batch mode. Interactive mode allows you to access the data directly and perform ad hoc queries against the database. Batch mode allows you to execute queries stored in a script file and save the results of those queries to a file.
mysqlaccess	Checks access privileges as they're configured in the grant tables of the mysql database.
mysqladmin	Provides an administrative interface for the MySQL installation. You can perform a variety of administrative tasks, such as obtaining information about the MySQL configuration, setting passwords, stopping the server, creating and dropping databases, and reloading privileges.
mysqlbinlog	Displays the binary update log file in a text format.
mysqlbug	Generates bug reports on Unix-like systems that you can then post to a MySQL mailing list.

Table continued on following page

Client program	Description
mysqlcheck	Checks and repairs MyISAM tables. You must use the mysqlcheck utility when the MySQL server is running, which is different from myisamchk, which you should not use when the server is running.
mysqldump	Copies the data in database tables into text files. This can be useful if you want to back up the data, create a test database, or move a database to another server.
mysqlhotcopy	Makes a quick backup of a database. It can run only on Unix-like systems and NetWare systems and must run on the same machine where the data directory is located.
mysqlimport	Copies data from a text file into tables in a MySQL database.
mysqlshow	Displays a list of the databases that currently exist in MySQL, a list of tables in a database, or information about a specific table.
perror	Displays a description of a system error code or of a table handler error code for MyISAM, ISAM, and DBD tables.

The programs listed in the table are not a complete list of all the client programs available in MySQL, although this list represents some of the more common ones. Be sure to check the MySQL product documentation for a complete list of programs, the options available for each program, and the platforms that support the programs. In addition, you can execute the program name and the `--help` option at a command prompt to learn more about that program.

The mysql Utility

One of the most useful programs included with MySQL is the mysql client utility that you use at a command prompt. The mysql utility allows you to perform a number of administrative tasks and to interact directly with the data stored in MySQL. You can use the mysql utility in interactive mode and in batch mode.

Using mysql in Interactive Mode

When you use the mysql utility in interactive mode, you launch the tool from a command prompt. For example, you can open the mysql utility in the context of the test database located on server1 as the root user. To do so, you would specify the following command:

```
mysql mysql -h server1 -u root -p
```

If you include these options in a configuration file, then you don't need to specify them here, unless you want to override the options as they're defined in the configuration file. If you don't need to override the options, then you can simply specify the program name, and the utility launches. When mysql opens, the command prompt changes to the mysql command prompt, as shown in the following:

```
mysql>
```

Once at the mysql command prompt you can execute SQL statements as well as commands specific to the mysql utility. For example, to execute a `SELECT` statement, you would simply type in a statement such as the following and then press Enter:

```
SELECT host, user FROM user;
```

The `SELECT` statement retrieves data from the host and user columns of the user table. When you execute the statement, you receive results similar to the following:

```
+-----------+-------+
| host      | user  |
+-----------+-------+
| %         | root  |
| localhost | admin |
| localhost | root  |
+-----------+-------+
3 rows in set (0.00 sec)
```

In this previous example, the `SELECT` statement is written on one line; however, a statement can span multiple lines. For example, you can enter the `SELECT` clause (the section of the statement that is introduced by the `SELECT` keyword) on one line, press Enter, type the `FROM` clause and the semi-colon, and then press Enter again. The statement is not processed until mysql sees the semi-colon.

The mysql client utility also allows you to execute commands that are specific to the mysql client utility. These commands are useful for performing such tasks as clearing your entry from the command prompt, exiting the mysql utility, and switching databases.

All mysql commands include a long form and a short form. For example, you can also write the prompt command as the `\R` command. You should note, however, that the short form is case sensitive. For instance, the `\R` command is the short form of the prompt command, but the `\r` command is the short form of the connect command.

To view a list of the mysql commands, type `help` at the mysql command prompt. You'll find that you'll need to use a number of these commands regularly. For these reasons, each command is examined individually. This section covers only the most commonly used mysql commands; for information about all the commands, see the MySQL product documentation.

> *When you execute a mysql command, you do not need to terminate the command with a semi-colon. Using a semi-colon, though, presents no problem.*

The clear Command

The clear (`\c`) command is quite useful if you decide part way through a statement that you want to discontinue the statement. For example, suppose you start a `SELECT` statement on one line and then press Enter, before you complete or terminate the statement. You will receive the following prompt:

```
->
```

The prompt indicates that mysql is waiting for you to complete the statement. To end the statement, you can enter `\c` and then press Enter, as shown in the following example.

```
SELECT host, user
\c
```

When you type the `\c` command and then press Enter, mysql clears the statement and then returns you to the mysql prompt, and no statements are executed.

With the clear command, you can use only the short form of the command to clear an SQL statement. All other mysql commands permit you to use the long form or the short form.

The exit and quit Commands

You should use the exit or quit (`\q`) command whenever you want to quit the mysql client utility. Simply type one of the commands at the mysql command prompt; then press Enter. For example, to quit mysql, type and enter the following command:

```
exit
```

You can also use the short form:

```
\q
```

When you execute the exit or quit command, the system returns you to your shell's command prompt, and your connection to the MySQL server is terminated.

The help Command

The help (`\?` or `?`) command displays a list of commands specific to the mysql utility. To use the help command, type it at the mysql command prompt, then press Enter. The commands then display.

The prompt Command

The prompt (`\R`) command allows you to change how the mysql prompt displays. For example, if you want the mysql prompt to display the name of the current database, you could type the following command:

```
prompt db:\d>
```

When you execute the command, the prompt changes to the following:

```
db:mysql>
```

The new prompt displays until you end the mysql session. If you want to use this prompt every time you use the mysql client utility, you should define the prompt option in the configuration file.

The prompt command supports a number of options that allow you to display the mysql prompt in various ways. The following table describes each of the options available to the prompt command. Note that these options are case sensitive and must be typed exactly as shown here.

Option	Description
\	Space (A space follows the backslash.)
_	Space
\\	Backslash
\c	A counter that increments by one for each statement that you execute (This option is not the same as the mysql clear (\c) command. The \c option shown here is specifically used as a switch for the prompt command.)
\d	Current database
\D	Current date
\h	Server host
\m	Minutes from current time
\n	New line (no prompt)
\o	Current month (numeric format)
\O	Current month (three-letter format)
\p	Current socket name, port number, or named pipe
\P	a.m./p.m.
\r	Current time (12-hour clock)
\R	Current time (24-hour clock)
\s	Seconds of the current time
\t	Tab
\u	Username
\U	Username and hostname
\v	Version of server
\w	Current day of the week (three-letter format)
\y	Current year (two-digit format)
\Y	Current year (three-digit format)

Any letter not used as a command, when preceded by a backward slash, is used literally. For example, \Big would be displayed as Big, but \Out would be displayed as <current month>ut, as in Decut.

If you execute the prompt command without any options, the default mysql command prompt displays.

The source Command

The source (\.) command executes a query that is in a specified file. For example, suppose you have a file named user_info.sql that contains a `SELECT` statement. You can execute the statement by using the source command to call the file, as in the following example:

```
\. c:\program files\mysql server <version>\user_info.sql
```

The source command accesses the user_info.sql file, which is stored in the `C:\Program Files\MySQL\MySQL Server <version>` directory, and executes the SQL statements in the file.

The status Command

The status (`\s`) command returns information about the current MySQL session, including the current user, database, connection ID, and server version. To use this command, type the command at the mysql command prompt and press Enter. The information then displays at the mysql command prompt.

The tee and notee Commands

The tee (`\T`) command is very useful if you want to keep a record of the commands that you type and the data that is returned by executing those commands. Using this command does not affect how data displays when you're viewing it interactively. It simply saves the input and output to a text file. For example, you can start logging your session by executing a command similar to the following:

```
tee c:\program files\mysql server <version>\shell.txt
```

A file named shell.txt is created if necessary and stored in the `C:\Program Files\MySQL\MySQL Server <version>` directory. You can stop logging to the file by simply executing the notee (`\t`) command. If you want to start logging your session again, simply reexecute the tee command. The new data is then appended to the existing text file.

The use Command

The use (`\u`) command allows you to change the current database. Whichever database you specify becomes the active database, assuming that you have the privileges necessary to access that database. For example, if you want to change the current database to the test database, you would use the following command:

```
use test
```

When you execute the command, the test database becomes the current database, and you can begin executing SQL statements against that database.

As you can see, mysql is a flexible tool that enables you to interact with MySQL data directly. You can perform ad hoc queries that allow you to administer MySQL and manage the data stored in the MySQL databases. The following Try It Out illustrates the mysql utility's flexibility.

Try It Out — Using the mysql Client Interactively

To use the mysql client utility interactively, take the following steps:

1. If necessary, open a shell for your operating system:

   If you're using a graphical user interface (GUI) such as GNOME or KDW in a Linux environment, use a Terminal window.

   If you're using Linux without a GUI, use the shell's command prompt.

   If you're working in a Windows environment, use a Command Prompt window.

2. Next, launch the mysql client utility in interactive mode. If you're using a Linux RPM installation of MySQL, enter the following command:

```
/usr/bin/mysql mysql -u root
```

If you're using a Linux tar installation of MySQL, enter the following command:

```
/usr/local/mysql/bin/mysql mysql -u root
```

If you're using a Windows installation of MySQL, enter the following command:

```
c:\program files\mysql\mysql server <version>\bin\mysql mysql -u root
```

You receive a message that welcomes you to the MySQL monitor and provides information about command termination, the MySQL connection, and getting help. In addition, the command prompt displays as follows:

```
mysql>
```

At the command prompt, you simply type SQL commands as you would type in commands in any Linux or Windows shell. When sending commands to MySQL, you must terminate your commands with a semi-colon, unless those commands are specific to the mysql utility, such as the exit or quit command.

3. Now, try out a couple of mysql-specific commands. In the first one, you list the commands that are specific to the mysql client. Execute the following command:

```
help
```

When you execute the help command, mysql returns a list of available commands, as shown in the following results:

```
For the complete MySQL Manual online visit:
   http://www.mysql.com/documentation

For info on technical support from MySQL developers visit:
   http://www.mysql.com/support

For info on MySQL books, utilities, consultants, etc. visit:
   http://www.mysql.com/portal

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear command.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set query delimiter.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
notee     (\t) Don't write into outfile.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute a SQL script file. Takes a file name as an argument.
```

```
status    (\s) Get status information from the server.
tee       (\T) Set outfile [to_outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.

For server side help, type 'help all'
```

The list of commands available to mysql can vary depending on version and operating system. Be sure to review the list that appears on your system.

4. Next, type in a `SELECT` statement, but don't include the terminator (semi-colon). Type the following at the command prompt and then press Enter:

```
SELECT host, user FROM user
```

Because you didn't include the terminator, mysql interprets this to mean that you have not completed the statement. As a result, the command prompt changes to the following:

```
->
```

At this point, you can complete the statement, add a semi-colon, or clear it to start over.

5. If you decide that you don't want to complete this statement, use the following command to clear it:

```
\c
```

The clear command tells mysql to disregard the statement and return you to the command prompt.

6. Another mysql command that you use quite often is the use command, which specifies the name of the database in which you'll be working. For example, you would change to the test database by using the use command:

```
use test
```

After you execute this command, you receive a message telling you that the database has been changed, and you're returned to the command prompt.

7. Finally, to exit the mysql utility, you can simply execute the following command:

```
quit
```

You could have also used the exit command or the `\q` command to exit mysql. Once you execute one of these commands, you're returned to your shell's command prompt.

How It Works

Once you're in your operating system's shell, you can launch the mysql client utility along with the options necessary to operate the utility. In the preceding exercise, you included two options, as shown in the following code:

```
/usr/bin/mysql mysql -u root
```

The first mysql refers to the client utility. If you want, you can launch the utility without specifying any options; however, if you do specify options, they follow the name of the utility. For example, the second mysql is an option that provides the name of a specific database, which in this case is the mysql administrative database that is installed by default as part of your MySQL installation. You can specify any

database that currently exists in your MySQL environment. By specifying the mysql database, you're telling mysql to operate in the context of that database. Once in the mysql shell, you can specify another database if you desire.

The second option used in the command is the user option: `-u root`. The option specifies that the mysql utility should interact with MySQL in the context of the root user account. Whatever privileges are granted to the specified user account are available during that mysql client session.

> *Because you haven't yet created any user accounts or modified the existing accounts, it isn't necessary to specify the root user. We used it here simply to demonstrate how that option is specified. Once you've created additional user accounts and assigned privileges to those accounts, you can log into the mysql client in the context of one of those accounts.*

After you logged into the mysql client utility, you were able to execute mysql commands, such as `help`, `\c`, `use`, and `quit`. These commands are specific to the mysql utility and, as a result, do not require that you terminate them with a semi-colon, as would be the case with an SQL statement. You can use a terminator, however, and the commands will run as they would without it. In addition, if you are executing several commands consecutively, as would be the case when creating a batch file, you must use the terminator so MySQL recognizes the end of one statement and the beginning of the next.

Using mysql in Batch Mode

Using the mysql client utility in batch mode provides you with a way to execute statements in a file from your shell's command prompt, without having to go into the mysql utility. To use the mysql client utility in batch mode, you must type the mysql command followed by the name of the source file, as shown in the following command:

```
mysql < <source file>
```

The source file can be any text file that contains SQL statements and mysql commands. If you execute this command, the results returned by the query are displayed at the command prompt. You can save those results to another file by adding that file to your command:

```
mysql < <source file> > <target file>
```

As you can see, you must define both a source file and a target file. Whenever you specify a file in this way, you must specify the appropriate path for each file.

As you have learned, the mysql utility is available not only in interactive mode, but in batch mode as well. The following Try It Out shows you how to use mysql in batch mode.

Try It Out — Using mysql in Batch Mode

To use the mysql client utility in batch mode, take the following steps:

1. Before you actually use mysql in batch mode, you need to create a file that contains an SQL statement. For this you need to use a text editor in order to create the file that includes the necessary commands. The new text file should contain the following statements:

```
use mysql;
SELECT host, user, select_priv FROM user;
```

Save the file as user_info.sql in a directory on your system.

2. If necessary, open a shell for your operating system:

   If you're using a graphical user interface (GUI) such as GNOME or KDW in a Linux environment, use a Terminal window.

   If you're using Linux without a GUI, use the shell's command prompt.

   If you're working in a Windows environment, use a Command Prompt window.

3. Next, launch the mysql client utility in batch mode. If you're using a Linux RPM installation of MySQL, enter the following command:

```
/usr/bin/mysql < <path>/user_info.sql
```

   The *<path>* placeholder refers to the path where you stored the user_info.sql file that you created in Step 1. If you're using a Linux tar installation of MySQL, enter the following command:

```
/usr/local/mysql/bin/mysql < <path>\user_info.sql
```

   If you're using a Windows installation of MySQL, enter the following command:

```
c:\program files\mysql\mysql server <version>\bin\mysql < <path>\user_info.sql
```

   When you execute on these commands, you receive the results of your `SELECT` statement without having to be in the mysql shell. These results are displayed in a manner similar to the following:

```
host    user    select_priv
localhost       root     Y
%       root    Y
```

   Of course, your exact results depend on whether you're working on a Linux or a Windows computer. The results displayed here are from a Windows computer.

This step provided the full directory path necessary to launch the mysql client program. The rest of this exercise assumes that you know how to launch the program from its correct path, so it provides only the command itself.

4. As you can see from these query results, they're much more difficult to read than if you were working directly in the mysql shell. You can improve the display by adding the `-t` option to you command:

```
mysql -t < <path>\user_info.sql
```

   The `-t` option, which you can also write as `--table`, returns the query in a tabular format, similar to what you would expect to see in the mysql shell, as shown by the following query results:

```
+-----------+-------+-------------+
| host      | user  | select_priv |
+-----------+-------+-------------+
| localhost | root  | Y           |
| %         | root  | Y           |
+-----------+-------+-------------+
```

   As you can see, this approach is far easier to read.

5. There might be times in which you don't want to view the results of your query, but instead want to save the results to a file. You can do this by modifying the command as follows:

```
mysql -t < <path>\user_info.sql > <path>\user_results.txt
```

When you execute this command, a file named user_results.txt is created in the specified directory. The new file contains the results of your query.

How It Works

Using mysql in batch mode allows you to execute SQL script files without having to interact directly with MySQL. As a result, before you can use mysql in batch mode, the source files have to exist. For this reason, you had to create the user_info.sql file to contain the following statements:

```
use mysql;
SELECT host, user, select_priv FROM user;
```

The first line in this statement instructs mysql to use the mysql database. The `use` command is a mysql command that allows you to specify a working database. The next line is a `SELECT` statement that retrieves information from the user table. Specifically, the statement returns data from the host, user, and select_priv columns in the user table.

Once you created the file, you were able to use the mysql utility to process the contents of the file, as shown in the following command:

```
mysql < <path>\user_info.sql
```

The command specifies the mysql client utility, followed by a left angle bracket, and the path and filename. The left angle bracket indicates to the mysql utility that it should be run in batch mode and that the file that follows the bracket contains the statements to be executed.

The statements contained in the file are executed, and the results are displayed at the command prompt. Whatever results you would normally expect to see when working with mysql interactively are displayed at the shell's command prompt. As a result, you did not have to launch mysql, type in the SQL statements, execute those statements, and then exit mysql. Instead, you were able to perform everything in one step, and you can rerun the query as often as you like, without having to retype it each time.

You then further refined the command by adding the `-t` option, which allowed the results to be displayed in the same format as you would see them if you executed the query while working in mysql interactively. You then refined your command even further by sending the query results to a separate file (user_results.txt), rather than displaying them at the command prompt, as shown in the following command:

```
mysql -t < <path>\user_info.sql > <path>\user_results.txt
```

The command specifies the mysql client utility, followed by a left angle bracket, and the path and filename. This is then followed by a right angle bracket and the path and filename of the target file. The right angle bracket indicates to the mysql utility that it should take the results returned by executing the statements in the first file and insert them into the second file.

By taking this approach, the query results are saved to a file, and you can view them or use the data as necessary, without having to run your query additional times. This is especially handy when working with large result sets or when you need to generate the same query numerous times on data that changes little or not at all.

Assigning Account Passwords

When you installed MySQL on Windows, you ran the MySQL Server Instance Configuration Wizard, which assigned a password to the MySQL root user account. When you installed Linux, though, you did not assign a password to the root account. As a result, anyone can sign into MySQL as the root user, without having to supply a password. As a result, one of your first steps in protecting MySQL should be to assign a password to root.

MySQL supports a couple ways to assign a password to a user account. The first method is to use the `mysqladmin` utility at your shell's command prompt. For example, suppose you want to assign a password to the myadmin account. You would use the following command:

```
mysqladmin -u myadmin password pw1
```

The command assigns the password pw1 to the account. From here on in, whenever that user connects to MySQL, the user must use the new password. If the account already has been assigned a password, you must know the current password when executing the command; in that case, your command might look like the following:

```
mysqladmin -u myadmin -p password pw1
```

When you execute this command, you are prompted for the current password, and then the account is assigned the new password. You can also assign a password from within the mysql client utility, in which case you would use a `SET PASSWORD` statement, as shown in the following example:

```
SET PASSWORD FOR 'myadmin'@'localhost' = PASSWORD('pw1');
```

Notice that you must also define the user and the host when assigning the password. The user must exist in order to use the `SET` statement to assign a password. If you're setting the password for an account that has multiple records in the user table, as is the case with the root account, you should use the `SET PASSWORD` statement, rather than the `mysqladmin` utility, and you should execute a `SET PASSWORD` statement for each record for that user, being certain to specify the correct computer in each case.

> *Whether you use the* `mysqladmin` *utility or a* `SET PASSWORD` *statement to assign a password to a user account, the password is saved to the user table in an encrypted format, and the user must supply the password when connecting to mysql.*

Once you set a password, you should reload the grant tables. Generally, it's a good idea to reload your grant tables into memory whenever you modify the tables in any way. Although it is not always necessary to reload the tables, it is not always clear under what circumstances they must be reloaded, so it's a good habit to foster. To reload the grant tables, run the following command at the mysql command prompt:

```
FLUSH PRIVILEGES;
```

Alternatively, you can run the `mysqladmin` utility at your shell's command prompt, as shown in the following command:

```
mysqladmin flush-privileges
```

Once the grant tables have been reloaded, the user must use a password to connect to MySQL.

As you have learned, one of the first steps that you should take before starting to use MySQL is to assign a password to the root user and to remove anonymous users. If you installed MySQL on Windows, you already assigned a password to the root user. If you installed MySQL on Linux, though, you still need to assign a password to the root user. The following Try It Out walks you through this process. If you're a Windows user, you can still work through this exercise.

Try It Out Securing MySQL Data

The following steps demonstrate how to assign a password to the MySQL root user account:

1. You create the necessary passwords from within the mysql utility. As usual, launch the utility from your shell's command prompt. This exercise assumes that you'll be launching the utility from the correct directory. Now open a shell window, and execute the following command at the command prompt:

```
mysql mysql
```

   The mysql utility launches, and the mysql database is the active database.

2. Next, you use the `SET PASSWORD` statement to assign a password to the root account. You need to replace the *<password>* placeholder in the following statement with a password of your choice. If you're a Windows user, you can use the same password that already exists, or you can assign a new password to the root account.

   At the mysql command prompt, execute the following command:

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('<password>');
```

   The `SET PASSWORD` statement uses the `PASSWORD( )` function to assign the password to the root user at the local computer.

3. Next, you must execute the `SET PASSWORD` statement a second time. For Linux installations, replace the *<computer>* placeholder with the name of your computer. For Windows installations, replace the *<computer>* placeholder with the percentage (%) wildcard. Now execute the following command:

```
SET PASSWORD FOR 'root'@'<computer>' = PASSWORD('<password>');
```

   The `SET PASSWORD` statement assigns the password to the second root record in the user table.

4. Next, you should reload the grant tables to ensure that MySQL is using the most current information. Execute the following command:

```
FLUSH PRIVILEGES;
```

   The `FLUSH PRIVILEGES` command reloads the changes that you just made. You might not always need to execute this command when updating the user table, but it's generally a good

idea in order to ensure that the system remains secure and that it behaves in the manner you would expect.

5. Now take a look at the changes that you just made to the user table. Execute the following command:

```
SELECT host, user, password FROM user;
```

On Linux, you should see results similar to the following:

```
+-----------+------+-------------------------------------------+
| host      | user | password                                  |
+-----------+------+----------------- -------------------------+
| localhost | root | *2B602296A79E0A8784ACC5C88D92E46588CCA3C3 |
| <name>    | root | *2B602296A79E0A8784ACC5C88D92E46588CCA3C3 |
| localhost |      |                                           |
| <name>    |      |                                           |
+-----------+------+-------------------------------------------+
4 rows in set (0.00 sec)
```

Instead of the *<name>* placeholder, you will see the name of your computer. If you decided to try out this exercise on a Windows installation, your results should be similar to the following:

```
+-----------+-------+-------------------------------------------+
| host      | user  | password                                  |
+-----------+-------+-------------------------------------------+
| localhost | root  | *2B602296A79E0A8784ACC5C88D92E46588CCA3C3 |
| %         | root  | *2B602296A79E0A8784ACC5C88D92E46588CCA3C3 |
+-----------+-------+-------------------------------------------+
2 rows in set (0.06 sec)
```

6. Next, exit the mysql utility. Execute the following command:

```
exit
```

You're returned to your shell's command prompt.

7. The next step is to start the mysql utility again. Only this time, you specify the user option, username, and password option, as shown in the following code:

```
mysql mysql -u root -p
```

When you execute this command, you'll be prompted for a password. Enter the password that you created earlier in this exercise, and then press Enter. You're then connected to MySQL as the root user.

8. Now that you've tested the new password, you can exit the mysql client utility and return to your operating system's command prompt.

How It Works

In this exercise, you used the mysql client utility to configure a password for the root user account, as shown in the following command:

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD('<password>');
```

As you can see, the command sets a password for root@localhost. As you'll recall, the user table contains two records for the root user. The first record is assigned to the localhost computer, and the second record is assigned to a host with the same name as your computer (on Linux) or assigned to all hosts (on Windows). As a result, you must execute the `SET PASSWORD` statement a second time:

```
SET PASSWORD FOR 'root'@'<computer>' = PASSWORD('<password>');
```

Although setting these passwords ensures that no one can log in as the root user without supplying the correct password, these changes don't prevent anonymous users from having at least some level of access to MySQL on Linux. In Chapter 14, you learn how to revoke permissions to users as well as remove users from the MySQL grant tables.

Setting Up a Configuration File

Once you assign a password to the root user account in MySQL and delete the anonymous accounts, you must include your username, password, and any other desired options whenever you connect to MySQL. This can result in a lot of retyping if you access the mysql client utility often. As you learned earlier in the chapter, though, you can place these options in a configuration file in the appropriate directory. That way, any time you connect as the root user, the options are called automatically from the configuration file.

If you're running MySQL on Linux, you need to create a configuration (option) file. The configuration file can be based on one of the four sample configuration files included with MySQL, or you can create your own. If you're running MySQL on Windows, you can copy the configuration file that was created when you first ran the MySQL Server Instance Configuration Wizard. After you copy the file, you can update it as necessary. To configure a configuration file for your MySQL installation, refer to one of the following two Try It Out sections. The first section covers Linux, and the second one covers Windows.

Try It Out — Setting Up a Configuration File in Linux

The following steps allow you to create a my.cnf configuration file and add the options necessary to connect to MySQL through the mysql utility:

1. Open a Terminal window (if you're working in a GUI environment) or use the shell's command prompt. In either case, you should be at the root directory.
2. You must copy the my-small.cnf configuration file to the root directory and rename it .my.cnf. For an RPM installation, execute the following command:

```
cp /usr/share/doc/packages/MySQL-server/my-small.cnf .my.cnf
```

   For a tar installation, execute the following command:

```
cp /usr/local/mysql/support-files/my-small.cnf .my.cnf
```

   The `cp` command copies the my-small.cnf file to the root directory and renames the file .my.cnf.

3. Once the file has been copied over, you can configure it as necessary for your MySQL programs. To edit the file, execute the following command:

```
vi .my.cnf
```

   The `vi` command opens the Vim text editor in Linux, which allows you to edit text files.

4. In order to edit the files, you must change the Vim utility to insert mode by typing the following letter:

```
i
```

   As soon as you type this command, you are placed in insert mode.

5. Scroll down to the section that begins with `[mysql]`, and add a blank line beneath the `[mysql]` heading.

6. Add the following code beneath the `[mysql]` heading:

```
user=root
password=pw1
database=mysql
```

   Be certain that you type the options correctly, or you will not be able to connect to MySQL. Notice the inclusion of the mysql database as the last option. As a result, whenever you start mysql, it opens in the mysql database. Later, as you're creating other databases, you might want to change this option or delete it entirely.

7. Press the Escape key to get out of Insert mode.

8. To indicate to the Vim utility that you have completed your edit, you must type the following command:

```
:
```

   When you type this command, a colon is inserted at the bottom of the screen and your cursor is moved to the position after the colon.

9. At the colon, type the following command and press Enter:

```
wq
```

   The `w` option tells the Vim utility to save the edits upon exiting, and the `q` option tells Vim to exit the program. When you execute this command, you're returned to the command prompt (at the root).

10. Next, open the mysql utility to ensure that you can connect to mysql without entering the required options. At the appropriate directory, execute the following command:

```
mysql
```

   You should be connected to MySQL, and the mysql prompt should be displayed.

11. Now type the following command to exit from mysql:

```
exit
```

   You're returned to your shell's command prompt.

How It Works

Setting up a configuration file on a Linux computer is a fairly straightforward process. You can use one of the sample files to create your configuration file, or you can create one from scratch. In this exercise, you copied the my-small.cnf sample file to your root directory and renamed it. From there, you edited the file so that it included the following options for the mysql client utility:

```
user=root
password=pw1
database=mysql
```

These options are the same options that you would enter into at the command prompt when you invoked the mysql utility; however, there are a few differences. For example, in the configuration file, you do not have to precede the `user` option or `password` option with hyphens. In addition, when you specify a database in the configuration file, you must precede the name of the database with the option name and an equal sign. When you specify the database option at a command prompt, you need only to specify the name of the database, and nothing else, as shown in the following command:

```
mysql mysql
```

Once you modify and save the configuration file, the options are used whenever you start mysql. As a result, you do not need to include any of the options that you defined in the configuration file, unless you want to override the options in the file. This next Try It Out covers how to set up a configuration file in a Windows environment.

Try It Out — Setting Up a Configuration File in Windows

The following steps allow you to set up the my.ini configuration file to include the options necessary to connect to MySQL:

1. Copy the `my.ini` file in `C:\Program Files\MySQL\MySQL Server <version>` to `C:\WINDOWS`.
2. Open the new file in Notepad or any text editor.
3. Scroll down to after the `[client]` section and before the `[mysqld]` section, and add the following `[mysql]` section:

```
[mysql]
user=root
password=<password>
database=mysql
```

   Be sure to replace the *<password>* placeholder with the password that you created for the root user account.

   Notice the inclusion of the mysql database as the last option. As a result, whenever you start mysql, it opens in the mysql database. Later, as you're creating other databases, you might want to change this option or delete it entirely.

4. Save and close the `my.ini` file.
5. Open a Command Prompt window, and change to the `C:\Program Files\MySQL\MySQL Server <version>\bin` directory. Execute the following command:

```
mysql
```

   Your username, password, and the mysql database are automatically used with this command. You can then use the mysql client utility from within the mysql database, or you can switch to another database.

6. Now type the following command to exit from mysql:

```
exit
```

You're returned to your shell's command prompt.

How It Works

In this exercise, you copied the `my.ini` file that was created when you ran the MySQL Server Instance Configuration Wizard. You then modified the file to include the following code:

```
[mysql]
user=root
password=pw1
database=mysql
```

The `[mysql]` heading indicates that the code following that heading applies specifically to the mysql utility. In this case, the options specify a user, password, and database. These options are the same options that you would use at the command prompt when invoking the mysql utility; however, there are a few differences. For example, in the configuration file, you do not have to precede the `user` option or `password` option with hyphens. In addition, when you specify a database in the configuration file, you must precede the name of the database with the option name and an equal sign. When you specify the database option at a command prompt, you need to specify only the name of the database, nothing else.

After you have modified and saved the configuration file, the options specified in the file will be used whenever you launch mysql. As a result, you will not need to specify any of the options that you defined in the configuration file, unless you want to override those options with new values.

Summary

This chapter provided a lot of detail about how to set up MySQL on your computer and find the programs that you use to interact with MySQL. The chapter also explained how to use the mysql client utility to work with data in a database. As a result, you are able to use this information to create databases and tables, add data to the tables, query and manipulate that data, and perform administrative tasks. You also learned how to secure your MySQL installation in order to avoid unauthorized users accessing your system. By the end of the chapter, you were provided with the background information and examples necessary to perform the following tasks:

- Work with the MySQL files as they are stored in your system's directories
- Work with the data directory and database files
- Work with the mysql administrative database
- Use the programs, scripts, and library files that are included with your MySQL installation
- Use the mysql client utility in interactive mode and in batch mode
- Assign passwords to MySQL user accounts
- Setting up your configuration file

Now that you are familiar with how to use MySQL and you have secured your installation, you're ready to add data and manipulate that data. Before you can store data in MySQL, though, the tables must exist that hold the data, and before the tables can exist, you must create the appropriate database to support those tables. Before you create a database, you should plan the structure of that database to ensure that it meets your needs as you begin adding data. As a result, the next step in working with MySQL is to learn how to design a relational database that you can implement in MySQL, which is what Chapter 4 covers.

Exercises

The following questions are provided as a way for you to better acquaint yourself with the material covered in this chapter. Be sure to work through each exercise carefully. To view the answers to these questions, see Appendix A.

1. You have used RPM files to install MySQL on Linux. You now want to launch the mysql client utility. In which directory will you find that utility?
2. You plan to use the mysql client utility to connect to MySQL. You will connect at a command prompt with the myadmin user account. You want to be prompted for your password when you launch the mysql utility. What command should you use to connect to MySQL?
3. You are modifying the configuration file on your system. You want the file to include options for the mysqladmin client utility. The file should specify that you're connecting to a host named system3 as a user named myadmin. The file should also specify the password as pw1. What code should you add to your configuration file?
4. You are working with a MySQL installation in Windows. You plan to use the mysql client utility in batch mode to run a query that has been saved to a file in the `C:\mysql_files` directory. The name of the file is users.sql, and it includes a command to use the mysql database, which the query targets. You want to save the results of the query to a file named users.txt, which should also be saved to the `C:\mysql_files` directory. What command should you use to execute the query?
5. You are using the mysql client utility to assign a password to the myadmin user account, which can access MySQL from any computer on your network. You want to assign the password pw1 to the user account. Which SQL statement should you use to assign the password?